



Writing better APIs with MicroProfile GraphQL

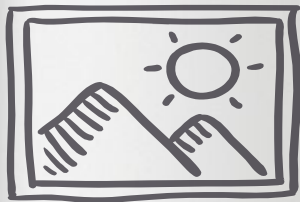
Phillip Krüger
April 2020



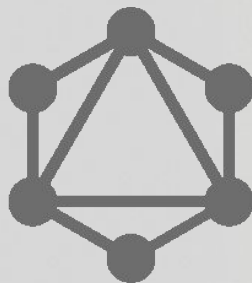
Red Hat

 @phillipkruger

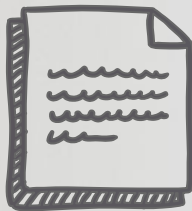
Story



GraphQL



Spec

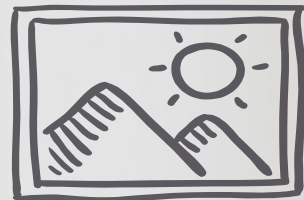


SmallRye



Future





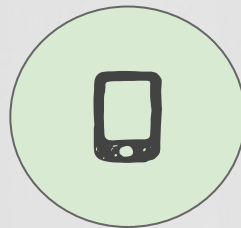
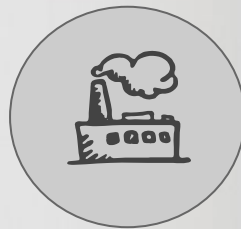
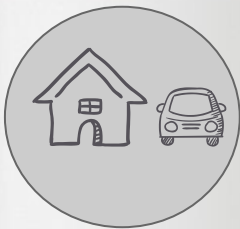
1. Painting a picture

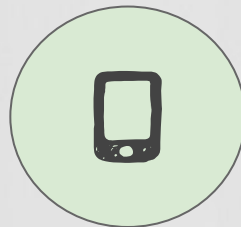
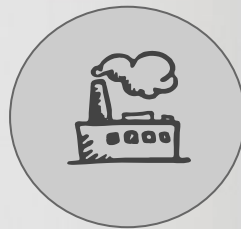
Giving you some context

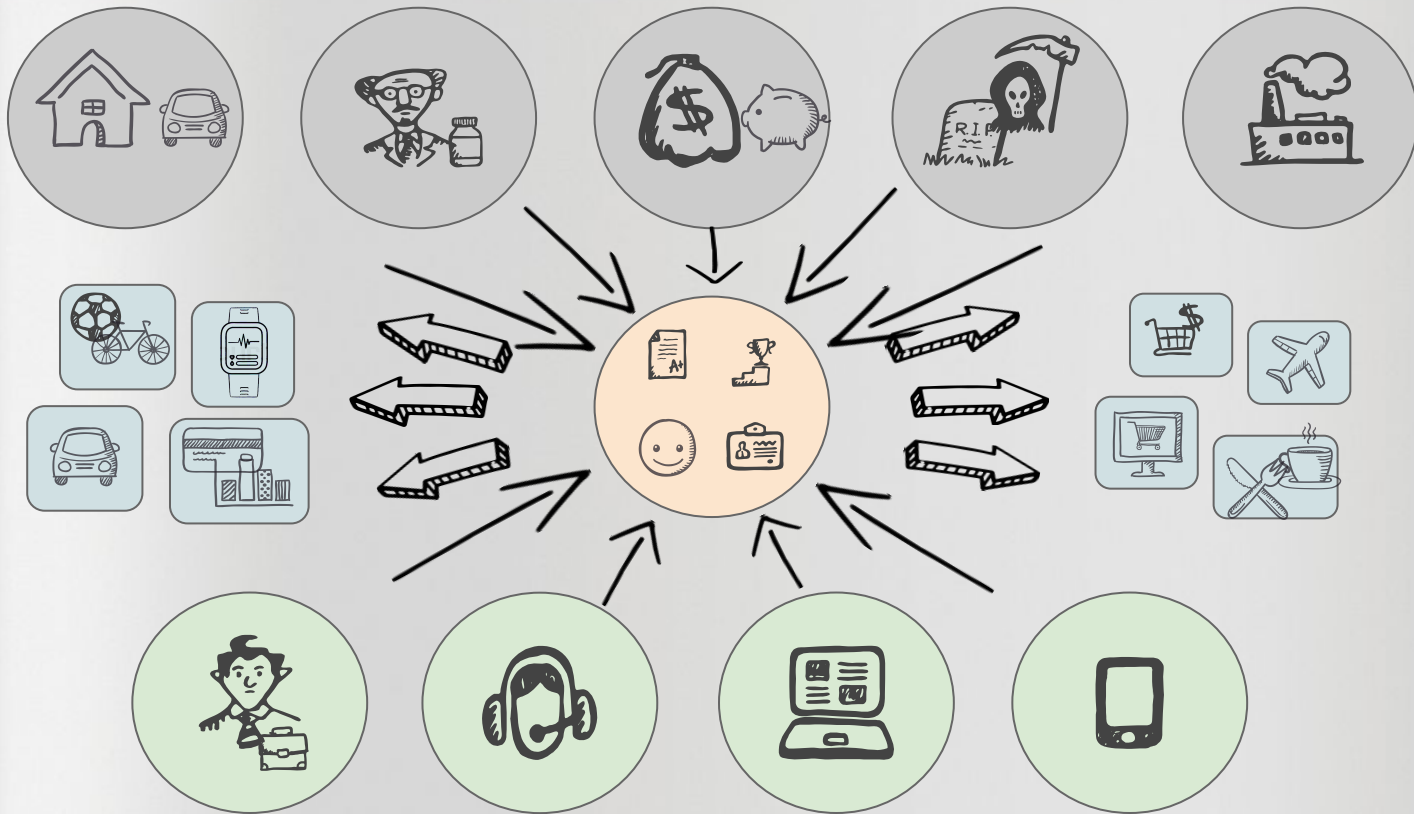




BIG COMPANY









Legacy

FTP



AS400

SOAP

RMI



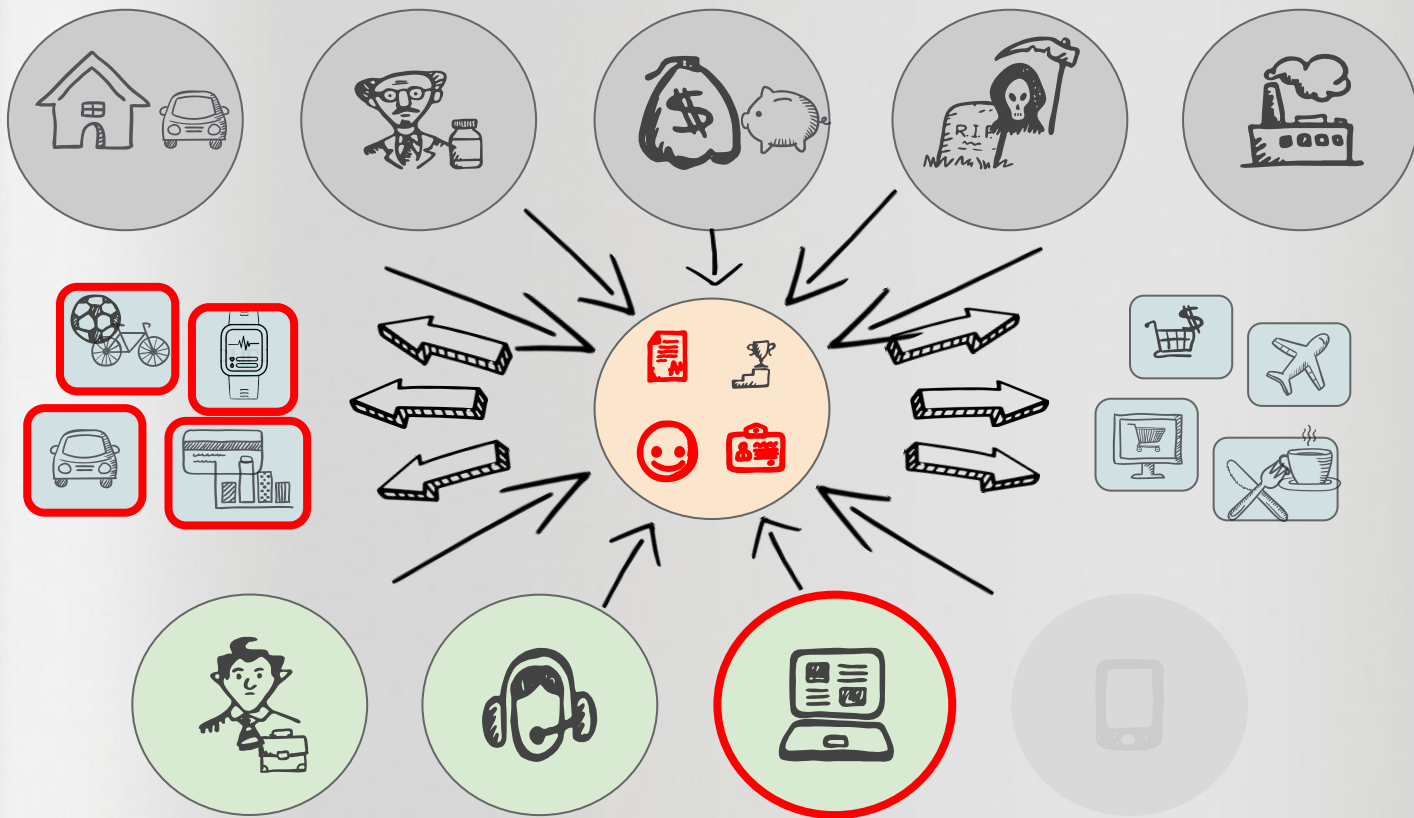
Java



REST



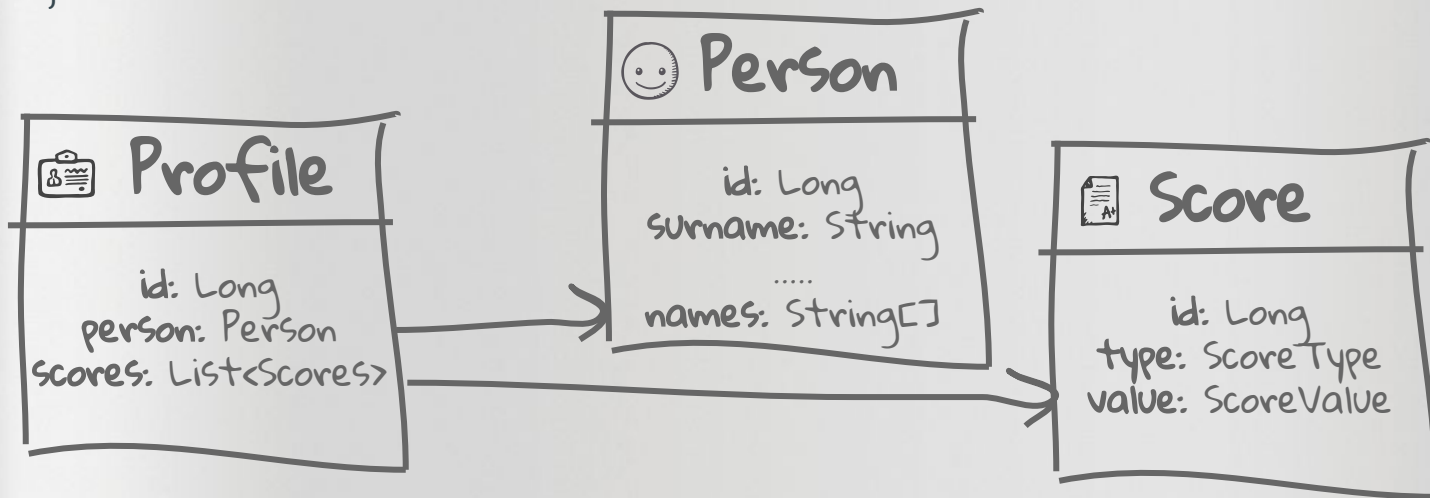
It's awesome ! Going to solve ALL our problems.



“

Ladies and gentlemen: the story you are about to hear is true. Only the names have been changed to protect the innocent..

```
@GET
@Path("/{userId}")
public Profile getProfile(@PathParam("userId") int userId){
    // Here find all data and return profile
}
```




```

{
  "id": 0,
  "person": {
    "addresses": [
      {
        "code": "86855-2543",
        "lines": [
          "28952 Warren Village",
          "North Rodrick",
          "New York",
          "Croatia"
        ]
      },
      {
        "code": "62388",
        "lines": [
          "49273 Ernsar Wells",
          "Kuhlmanfort",
          "Georgia",
          "Djibouti"
        ]
      }
    ],
    "biography": "Delenti a ut unde perspicatis qui est tempora. Velit asperiores et. Ex numquam quos reiciendis  

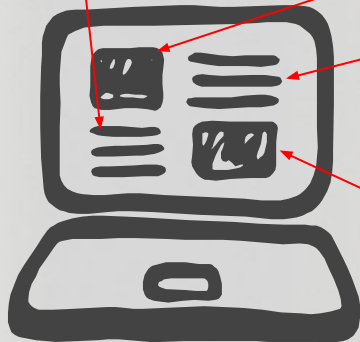
    cecet vitae sint repudiande.  

    Mione molestiae delectus natus consequatur. Est a molestias nostrum rerum et odio blanditiis. Veritatis ducimus et  

    consequatur modi.  

    Necessitatibus aut rerum. Distinctio debitis quidem eum beatae officis.",
    "birthdate": "77/18/1909",
    "coverphotos": [
      {
        "http://lorempixel.com/g/1680/1280/business/"
      }
    ],
    "creditCards": [
      {
        "expiry": "2011-10-12",
        "number": "1234-2121-1221-1211",
        "type": "mastercard"
      }
    ],
    "emailAddresses": [
      {
        "email": "macy.frisch@gmail.com",
        "chadwick.keeble@yahoo.com"
      }
    ],
    "favColor": "ivory",
    "gender": "Female",
    "id": 1,
    "idNumber": "882-18-8226",
    "inclients": [
      {
        "identifier": "Slack",
        "is": "cole.waelchi"
      },
      {
        "identifier": "IOU",
        "is": "barry.bergnum"
      }
    ],
    "interests": [
      "Overwatch",
      "cs:go"
    ],
    "joinDate": "24/09/2017",
    "locale": "en-ZA",
    "maritalStatus": "Divorced",
    "names": [
      "Alexandra",
      "Leonia"
    ],
    "nicknames": [
      "Annie Moore"
    ]
  }
}

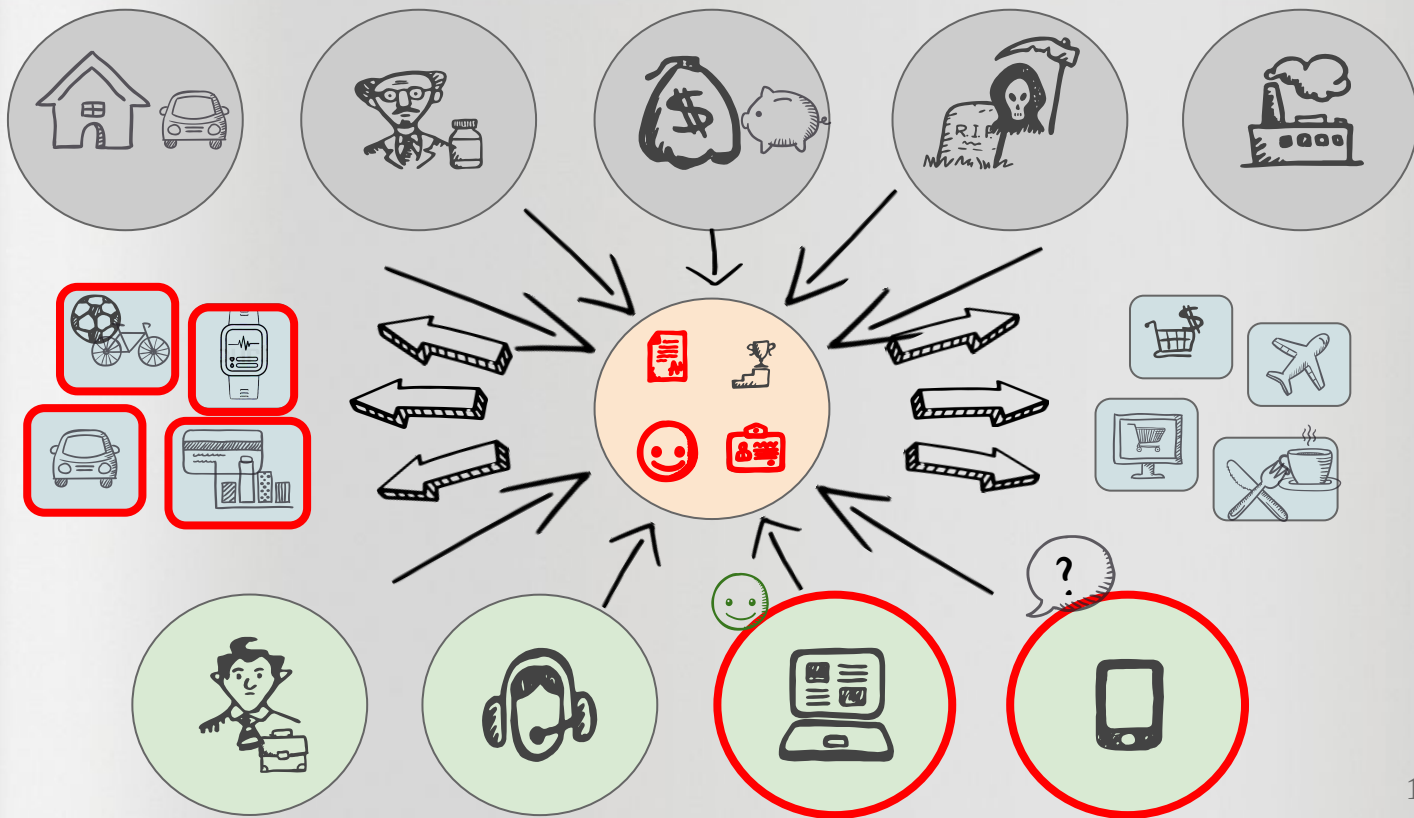
```



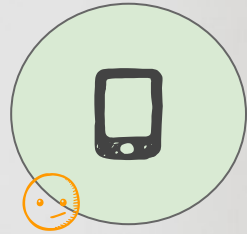
```

    "occupation": "Producer",
    "organization": "Jerde-Kertzmann",
    "phoneNumbers": [
      {
        "number": "158-692-8487",
        "type": "Cell"
      },
      {
        "number": "387.717.3673 x798",
        "type": "Home"
      },
      {
        "number": "1-833-318-6728",
        "type": "Work"
      }
    ],
    "profilePictures": [
      {
        "https://i3.amazonaws.com/u/faces/faces/twitter/ema/128.jpg"
      }
    ],
    "relations": [
      {
        "personURI": "rest/person/788",
        "relationship": "Spouse"
      }
    ],
    "skills": [
      "Networking skills",
      "Fast learner"
    ],
    "socialMedias": [
      {
        "name": "Pflorentino.bahringer",
        "username": "Twitter"
      },
      {
        "name": "ryan.christiansen",
        "username": "Facebook"
      }
    ],
    "surname": "Gleichner",
    "taglines": [
      "The dark side clouds everything. Impossible to see the future is.",
      "Chuck Norris burst the dot com bubble."
    ],
    "title": "Miss",
    "userAgent": "Opera/9.80 (X11; Linux i686; Ubuntu/14.10) Presto/2.12.388 Version/12.16",
    "username": "nolan.roberts",
    "websites": [
      "http://www.burt-zieme.io",
      "http://www.artie-windler.co"
    ]
  },
  "scores": [
    {
      "name": "Driving",
      "value": 8
    },
    {
      "name": "Fitness",
      "value": 8
    },
    {
      "name": "Activity",
      "value": 77
    },
    {
      "name": "Financial",
      "value": 15
    }
  ]
}

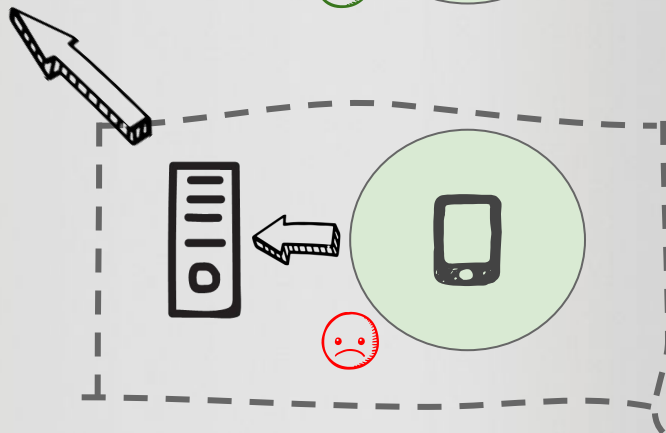
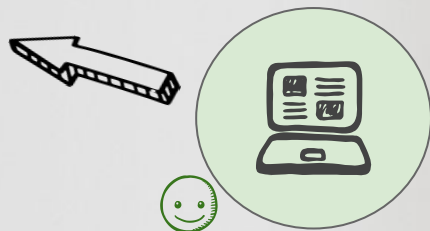
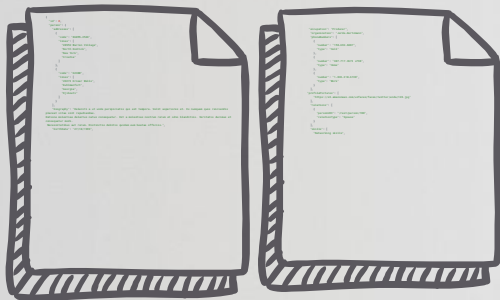
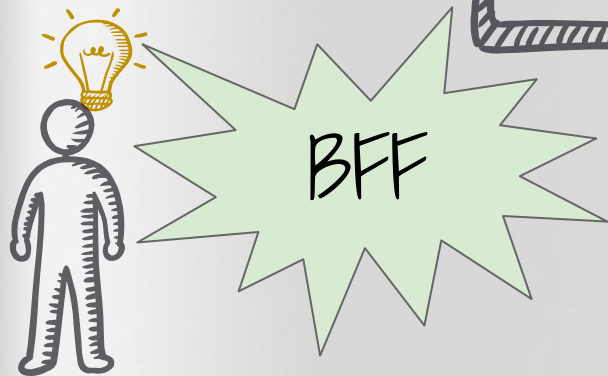
```




```
{  
  "id": "062-42-0952",  
  "person": "/rest/person/1",  
  "scores": "/rest/score/062-42-0952"  
}
```

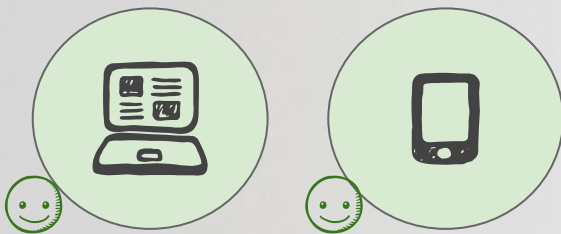


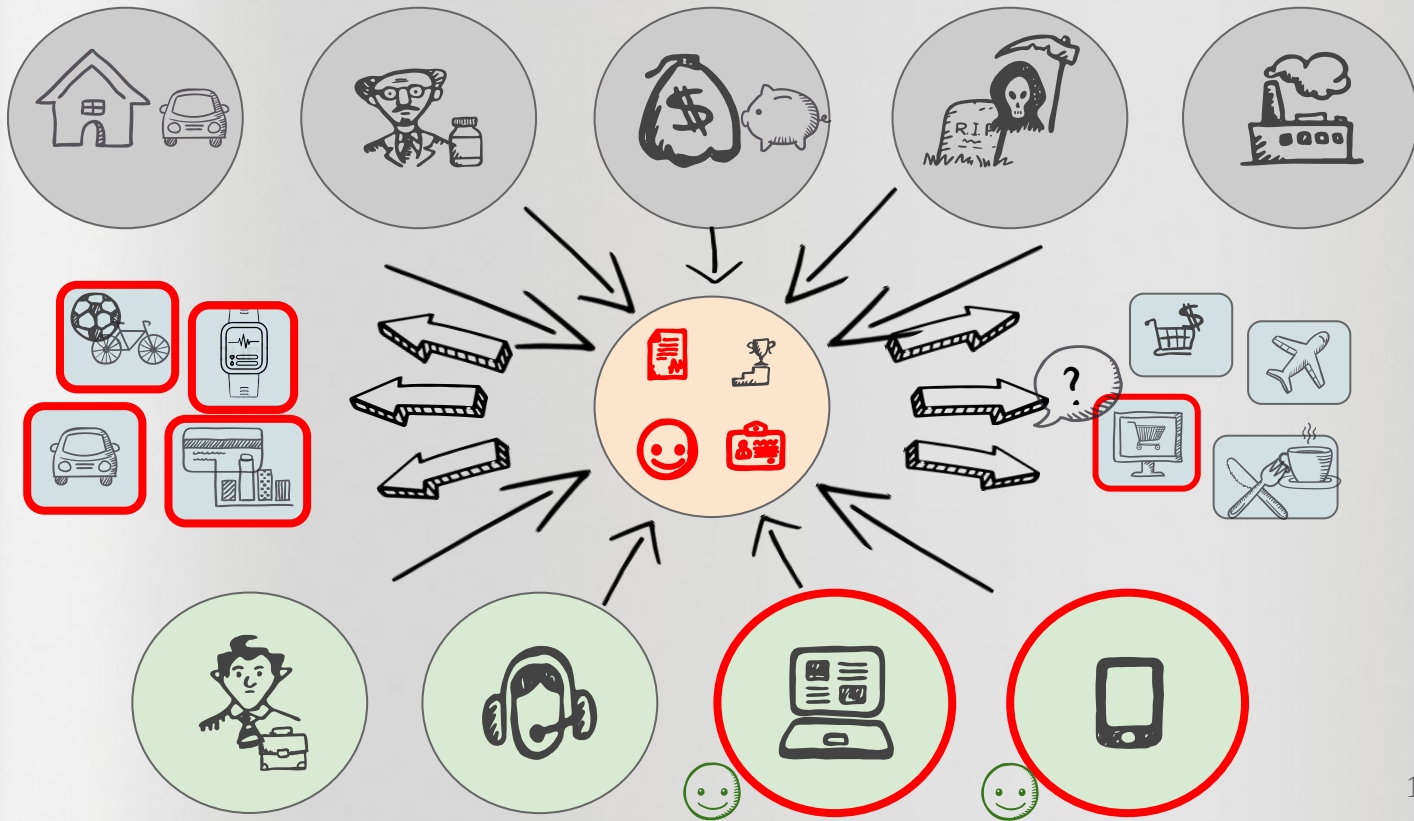
It's awesome ! Going to solve ALL our problems.



```
@GET
@Path("/{userId}")
public Profile getProfile(@PathParam("userId") int userId){
    // Here find all data and return profile
}

@GET
@Path("/lite/{userId}")
public ProfileLite getLiteProfile(@PathParam("userId") int userId){
    // Here find some data and return profile
}
```






```
@GET
@Path("/{userId}")
public Profile getProfile(@PathParam("userId") int userId){
    // Here find all data and return profile
}
```

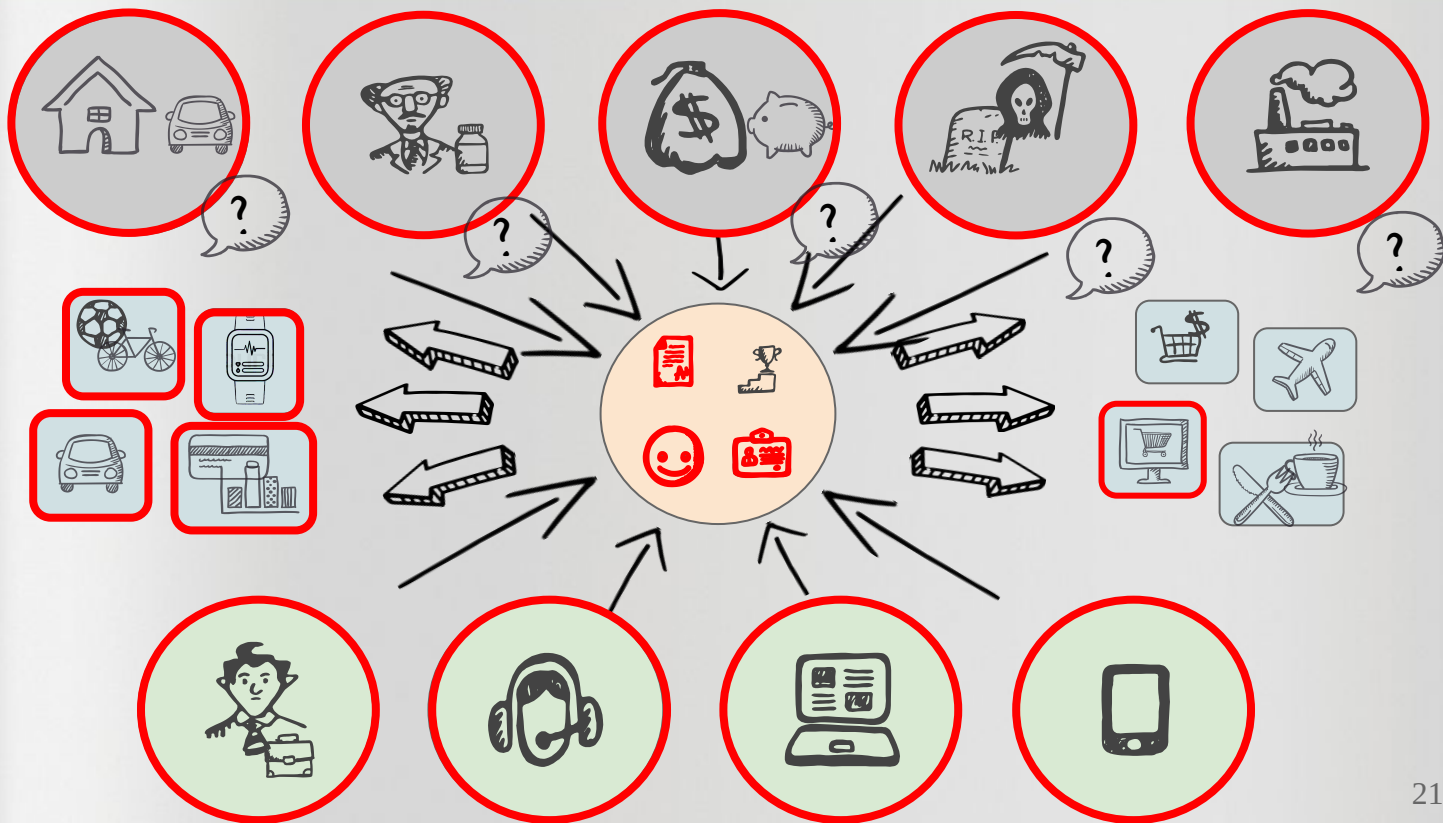
```
@GET
@Path("/lite/{userId}")
public ProfileLite getLiteProfile(@PathParam("userId") int userId){
    // Here find some data and return profile
}
```

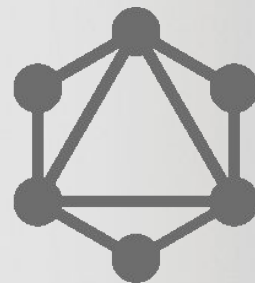
```
@GET
@Path("/lite/status/{userId}")
public ProfileLiteWithStatus
getLiteProfileWithStatus(@PathParam("userId") int userId){
    // Here find some data (and the status) and return profile
}
```

```
@GET
@Path("/{userId}")
public Profile getProfile(@PathParam("userId") int userId){
    // Here find all data and @QueryParam("include") String[] include,
    }
    @QueryParam("exclude") String[] exclude){
    // Here find all data and return profile
    }
}
```

“ **Over-fetching** is fetching too much data, aka there is data in the response you don't use.

Under-fetching is not having enough data with a call to an endpoint, leading you to call a second endpoint.





2. *Back to the* **Drawing board**

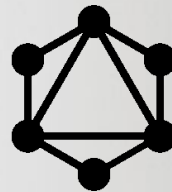
Surely someone else have these issues ?

Options

- ◆ HATEOAS
- ◆ BFF



Queryable REST



GraphQL

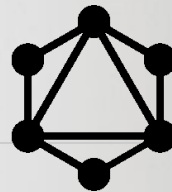
It's awesome ! Going to solve ALL our problems.



History

- ◆ Developed and open sourced by Facebook
- ◆ Specification <http://facebook.github.io/graphql>
- ◆ Alternative to REST
- ◆ Declarative data fetching
- ◆ Increased mobile usage
- ◆ Variety of different frontend frameworks
- ◆ Rapid feature development
- ◆ Since 2012. Publically 2015

Some benefits



Data Fetching

No more Over- and Under Fetching

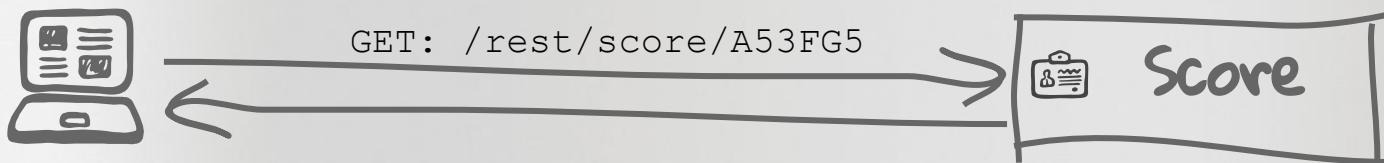
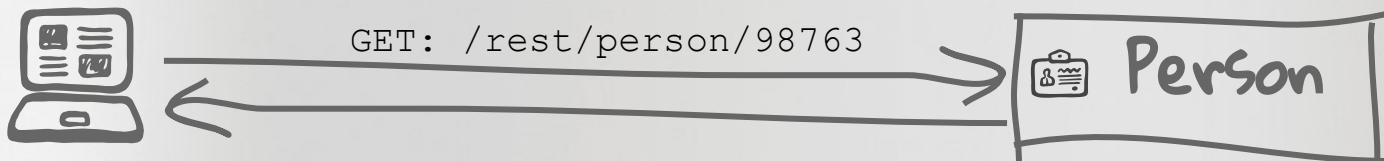
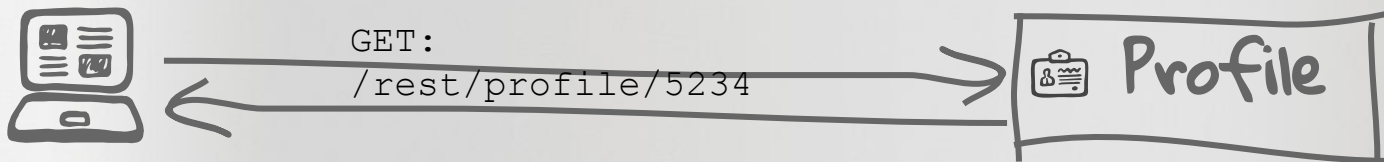
Development Speed

Rapid Product Iterations on the Frontend

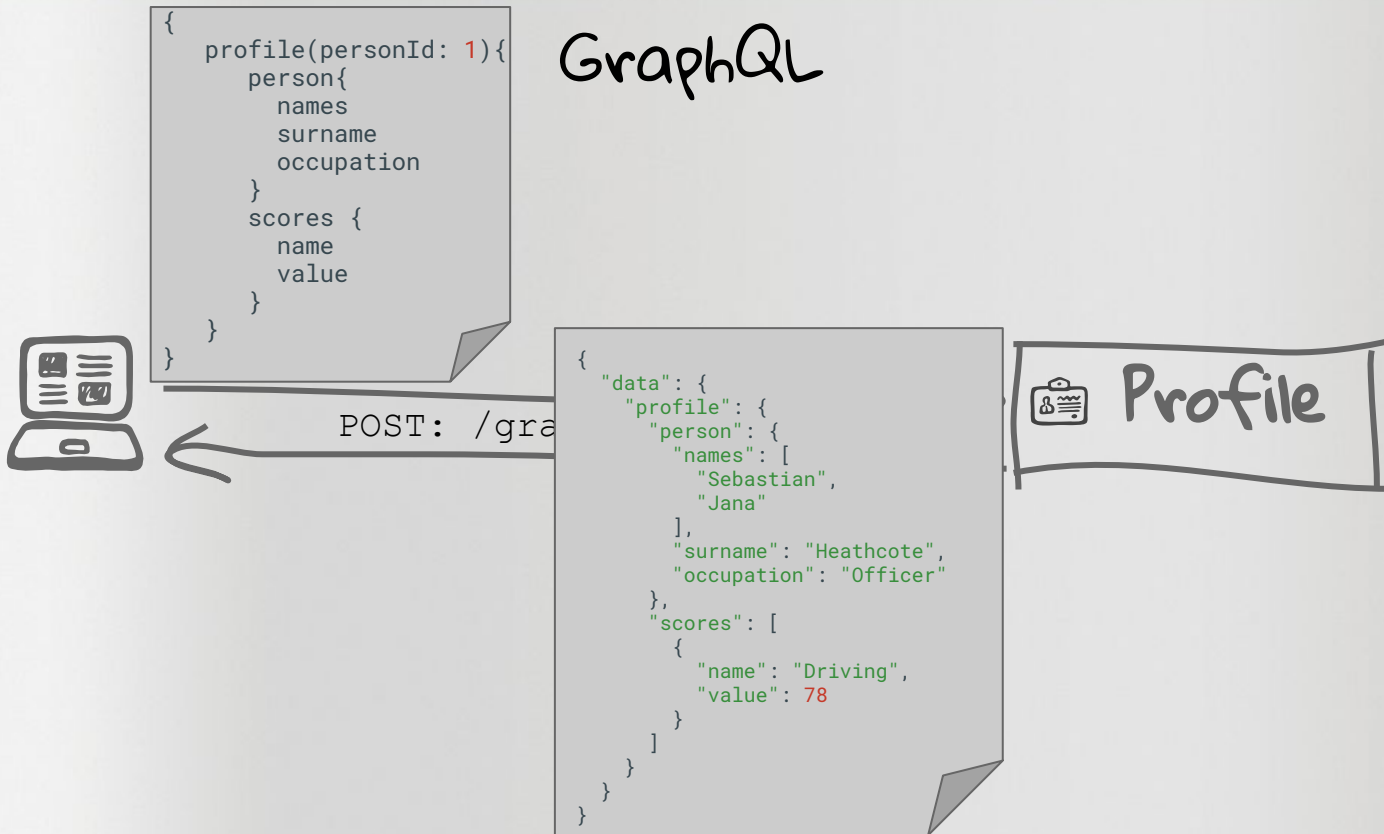
Schema

Benefits of a Schema & Type System

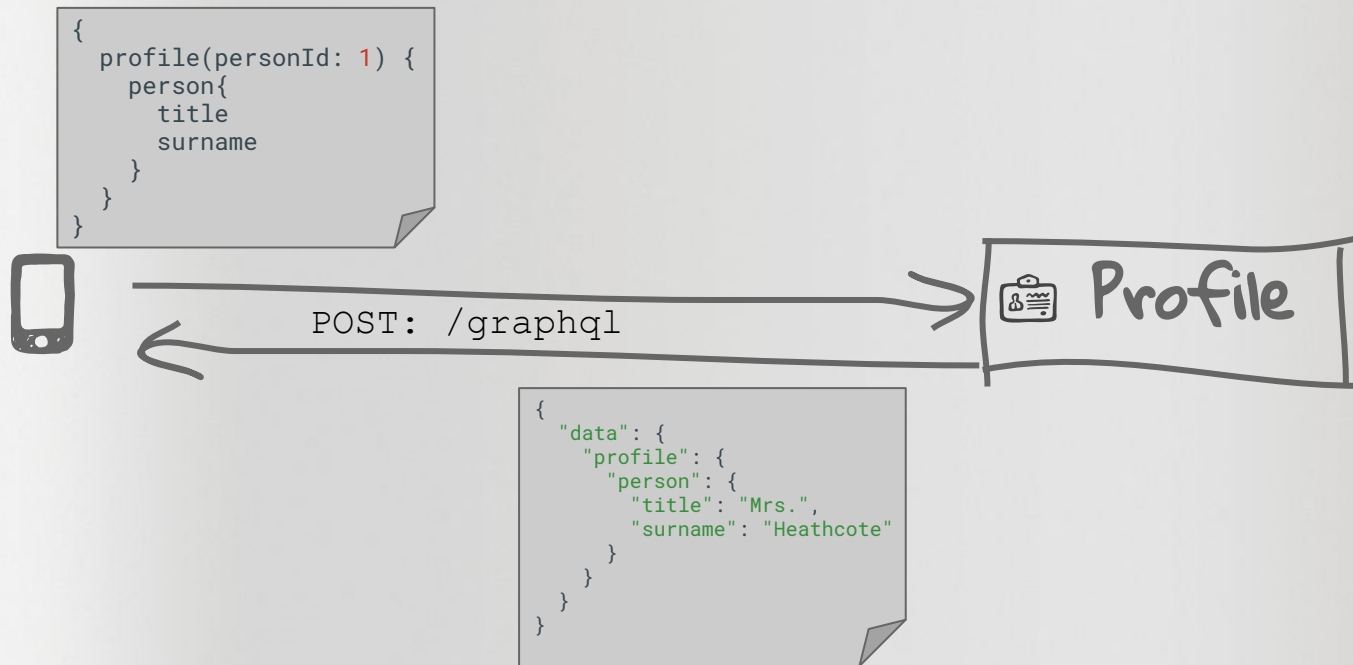
REST



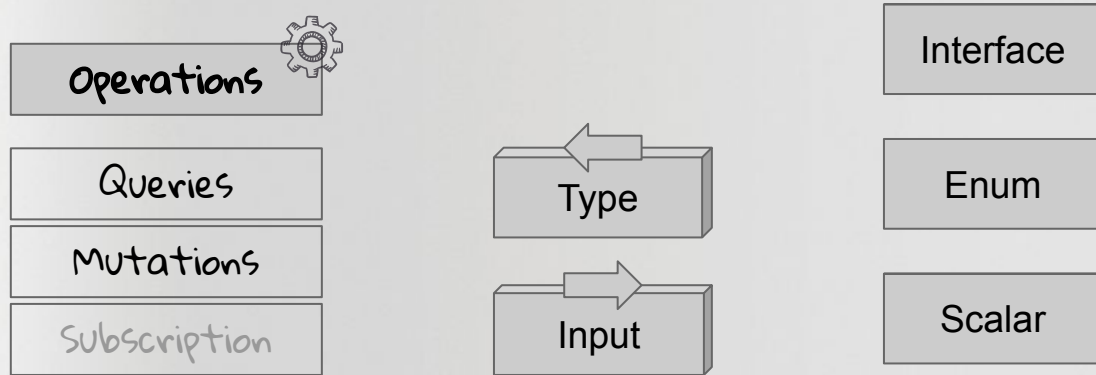
GraphQL



GraphQL



GraphQL Concepts



GraphQL Schema

```
type Address {
  code: String
  lines: [String]
}

type CreditCard {
  expiry: String
  number: String
  type: String
}

type ImClient {
  identifier: String
  im: String
}

#Mutation root
type Mutation {
  deletePerson(id: Int!): Person
  updatePerson(person: PersonInput): Person
}

type Person {
  addresses: [Address]
  biography: String
  #dd/MM/yyyy
  birthDate: Date
  coverphotos: [String]
  creditCards: [CreditCard]
  emailAddresses: [String]
```

```
type Profile {
  id: String
  person: Person
  scores: [Score]
}

#Query root
type Query {
  allScoreValues: [[Score]]
  allScoresKeys: [String]
  people: [Person]
  person(personId: Int!): Person
  personsWithSurname(surname: String = "Kruger"):
  [Person]
  profile(personId: Int!): Profile
  profileFull(personId: Int!): Profile
}

input CreditCardInput {
  expiry: String
  number: String
  type: String
}

type SocialMedia {
  name: String
  username: String
}

enum Gender {
  Female
  Male
}
```

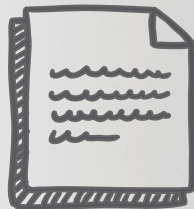


GraphQL Introspection

```
{
  __schema {
    types {
      name
      description
      kind
    }
  }
}
```

```
{
  "data": {
    "__schema": {
      "types": [
        {
          "name": "Address",
          "description": null,
          "kind": "OBJECT"
        },
        {
          "name": "AddressInput",
          "description": null,
          "kind": "INPUT_OBJECT"
        },
        {
          "name": "Boolean",
          "description": "Built-in Boolean",
          "kind": "SCALAR"
        },
        {
          "name": "CreditCard",
          "description": null,
          "kind": "OBJECT"
        },
        {
          "name": "CreditCardInput",
          "description": null,
          "kind": "INPUT_OBJECT"
        },
        {
          "name": "Date",
          "description": "Scalar for Date",
          "kind": "SCALAR"
        }
      ]
    }
  }
}
```





3. Specification

Open source community specification for Enterprise Java

MicroProfile






Optimizing Enterprise Java for a Microservices Architecture



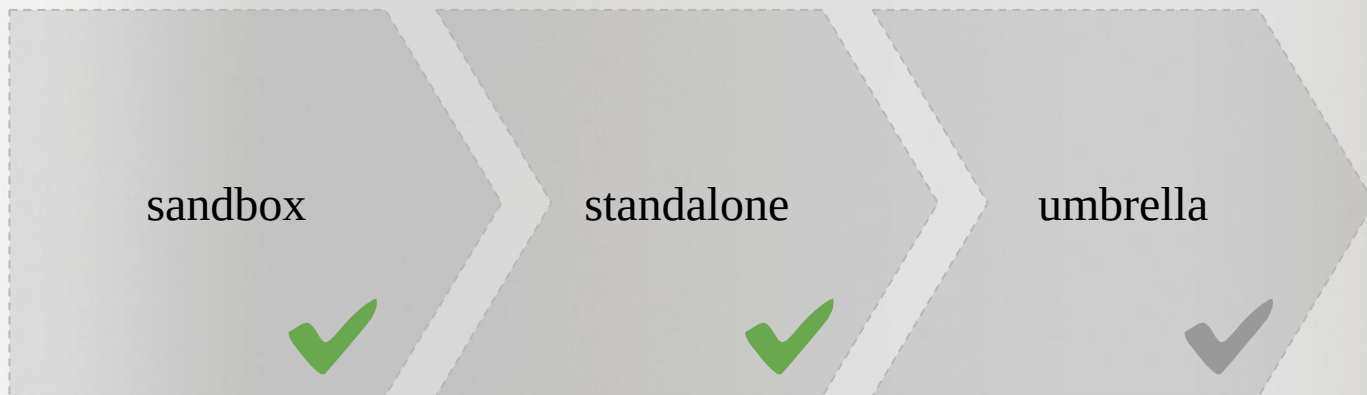
Open Tracing 1.3	Open API 1.1	Rest Client 1.3	Config 1.3
Fault Tolerance 2.0	Metrics 2.2	JWT Propagation 1.1	Health 2.1
CDI 2.0	JSON-P 1.1	JAX-RS 2.1	JSON-B 1.0

MicroProfile 3.2

 = New
 = Updated
 = No change from last release (MicroProfile 3.1)



MicroProfile Process





MicroProfile Process

<https://github.com/eclipse/microprofile-graphql>



Spec
(AsciiDoc)



API
(Java interfaces
& Annotations)



TCK
(TestNG)



CI
(Java)



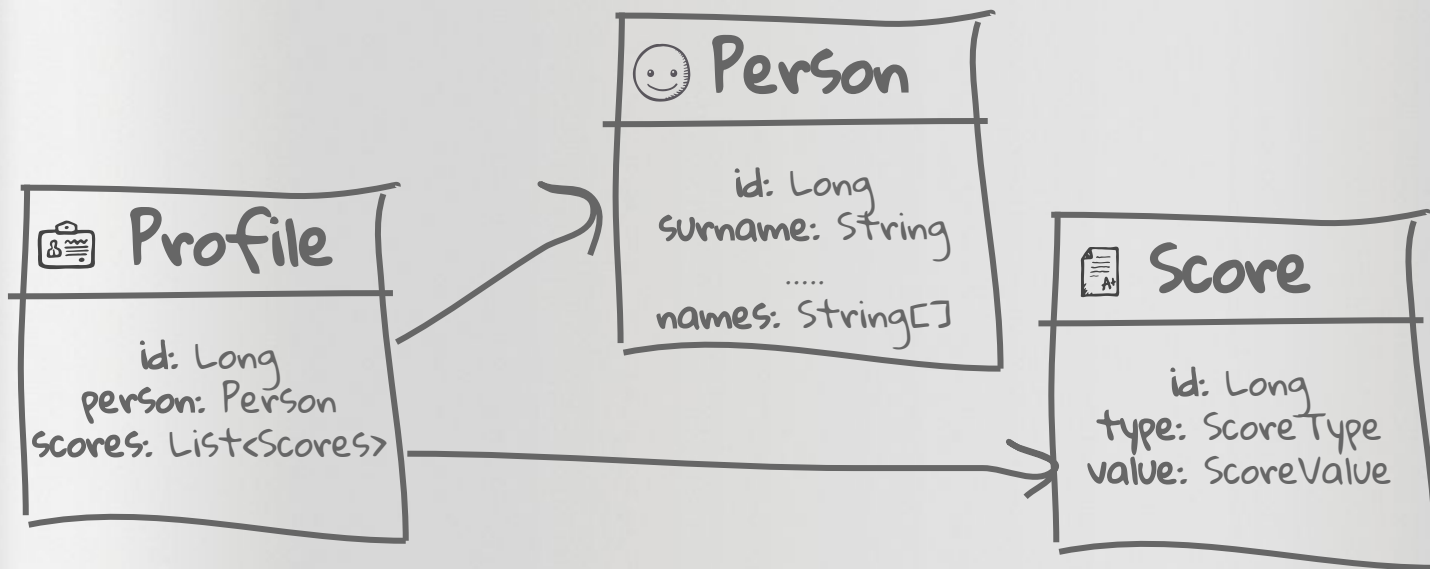
4. Examples

Using SmallRye GraphQL

<https://github.com/phillip-kruger/graphql-example>



Demo 1: Java EE / Jakarta EE Jax-RS



Demo 1: Java EE / Jakarta EE Jax-RS

```
@GET
@Path("/{personId}")
@Operation(description = "Get a person's profile using the person's Id")
public Profile getProfile(@PathParam("personId") int personId){

    Person person = personDB.getPerson(personId);
    List<Score> scores = scoreDB.getScores(person.getIdNumber());

    Profile profile = new Profile();
    profile.setId(person.getIdNumber());
    profile.setPerson(person);
    profile.setScores(scores);

    return profile;
}
```

Demo 1: MicroProfile GraphQL

```
@Query
@Description("Get a person's profile using the person's Id")
public Profile getProfile(int personId) {

    Person person = personDB.getPerson(personId);
    List<Score> scores = scoreDB.getScores(person.getIdNumber());

    Profile profile = new Profile();
    profile.setId(person.getIdNumber());
    profile.setPerson(person);
    profile.setScores(scores);

    return profile;
}
```



Demo 1

<https://github.com/phillip-kruger/graphql-example>

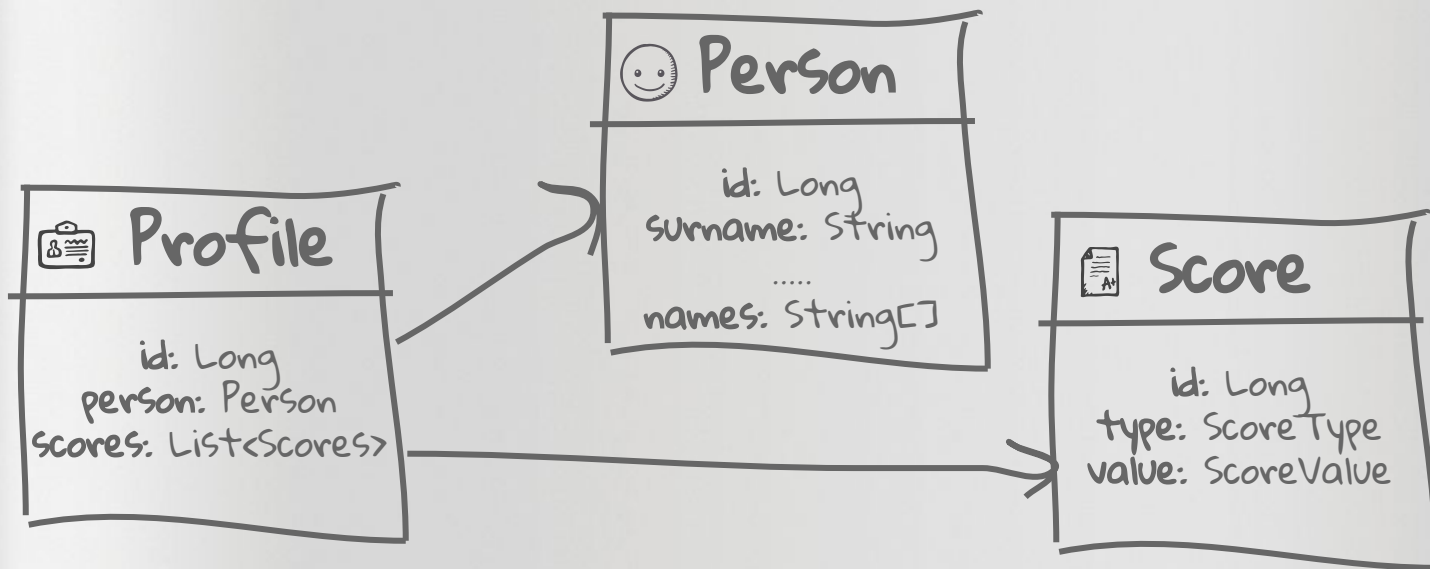
41



Red Hat

 @phillipkruger

Demo 2: MicroProfile GraphQL



Demo 2: MicroProfile GraphQL

```
@Query
public Profile profile(int personId) {
    Person person = personDB.getPerson(personId);
    //List<Score> scores = scoreDB.getScores(person.getIdNumber());
    Profile profile = new Profile();
    profile.setId(person.getIdNumber());
    profile.setPerson(person);
    //profile.setScores(scores);
    return profile;
}

public List<Score> scores(@Source Profile profile) {
    Person person = profile.getPerson();
    return scoreDB.getScores(person.getIdNumber());
}
```



Demo 2

<https://github.com/phillip-kruger/graphql-example>

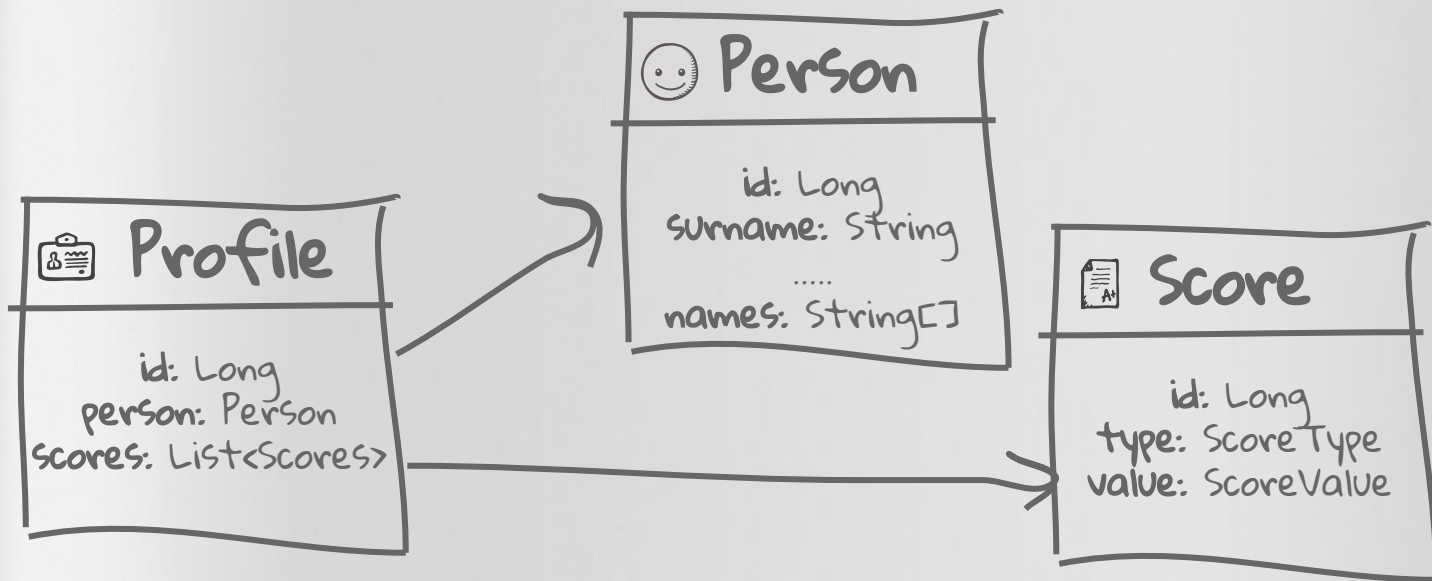
44



Red Hat

 @phillipkruger

Demo 3: MicroProfile GraphQL



Demo 3: MicroProfile GraphQL

```
@Query
public Person person(int personId){
    return personDB.getPerson(personId);
}

public List<Score> scores(@Source Person person) {
    return scoreDB.getScores(person.getIdNumber());
}
```



Demo 3

<https://github.com/phillip-kruger/graphql-example>

47



Red Hat

 @phillipkruger

Demo 4: Query Collections

@Query

```
public List<Person> getPeople(){  
    return personDB.getPeople();  
}
```





Demo 4

<https://github.com/phillip-kruger/graphql-example>

49



Red Hat

 @phillipkruger

Demo 5: Mutations

`@Mutation`

```
public Person updatePerson(Person person){  
    return personDB.updatePerson(person);  
}
```

`@Mutation`

```
public Person deletePerson(int id){  
    return personDB.deletePerson(id);  
}
```




Demo 5

<https://github.com/phillip-kruger/graphql-example>

51



Red Hat

 @phillipkruger

Demo 6: Errors and partial responses

```
{  
  people{  
    surname  
    scores{  
      thisDoesNotExist  
    }  
  }  
}
```

Demo 6: Errors and partial responses

```
public List<Score> scores2(@Source Person person)
    throws ScoresNotAvailableException {
    // Simulate failure
    throw new ScoresNotAvailableException(
        "Scores for person [" + person.getIdNumber() + "] is not
        available");
}
```




Demo 6

<https://github.com/phillip-kruger/graphql-example>

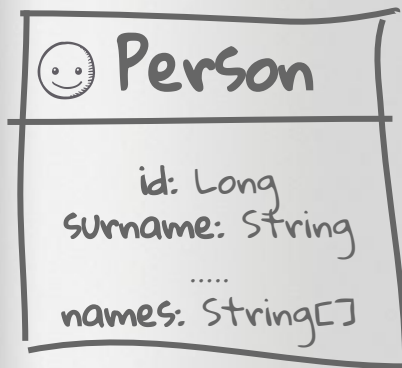
54



Red Hat

@phillipkruger

Demo 7: More complex graphs



Demo 7: More complex graphs

```
@Query
public Person person(int personId){
    return personDB.getPerson(personId);
}

public List<Score> scores(@Source Person person) {
    return scoreDB.getScores(person.getIdNumber());
}

public List<Event> events(@Source Score score) {
    return eventDB.getEvents(score.getId());
}
```




Demo 7

<https://github.com/phillip-kruger/graphql-example>

57



Red Hat

 @phillipkruger

Demo 8: JsonB Support and other annotations

```
public class Event {  
    private Action action;  
    private BigDecimal value;  
    private LocalDateTime dateTime;  
    @JsonbDateFormat("dd MMM yyyy 'at' HH:MM")  
    private LocalDateTime when;  
}
```



Demo 8

<https://github.com/phillip-kruger/graphql-example>

59



Red Hat

 @phillipkruger

Demo 8: JsonB Support

```
@JsonbProperty("monthlySalary")  
@JsonbNumberFormat(value = "¤ 000.00", locale = "en_ZA")  
private BigDecimal salary;  
  
@JsonbTransient  
private BigDecimal bonus;
```

Field / Getter / Setter

Demo 8: Other annotations

```
@Name("monthlySalary")  
@NumberFormat(value = "¤ 000.00", locale = "en_ZA")  
private BigDecimal salary;
```

```
@Ignore  
private BigDecimal bonus;
```

```
@DateFormat("MM/dd/yyyy")  
private LocalDate usDate;
```

Field / Getter / Setter

Demo 8: Other annotations

```
private List<@NumberFormat("α 000.00") BigDecimal> transactions;
```

@NonNull

```
private String surname;
```

```
private List<@NonNull String> names;
```

@Query

```
public List<Person> withSurname(@DefaultValue("Kruger") String surname) {
```


Demo 8: Other annotations

@Type

@Input

@Enum

@Interface

@Source



Demo 9: Introspection

```
{  
  __schema{  
    types {  
      name  
      kind  
    }  
  }  
}
```





Demo 9

<https://github.com/phillip-kruger/graphql-example>

65



Red Hat

 @phillipkruger



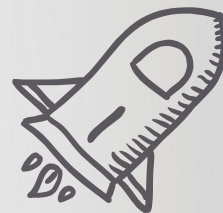
Known Implementations



<https://github.com/smallrye/smallrye-graphql>

SPQR

<https://github.com/leangen/graphql-spqr>



5. What's Next ?

What we are working on now

Disclaimer: Work in progress !

Client

Low level

Dynamic

Fluent

???



```
Builder builder = new Builder(Operation.Type.QUERY)
    .addRootField("allHeroesIn",
        args(
            arg("city", "New York, NY")
        ),
        fields(
            field("name"),
            field("currentLocation"),
            field("teamAffiliations",
                fields(
                    field("name")
                )
            )
        )
    );
```

```
query {
  allHeroesIn(city: "New York"){
    name
    currentLocation
    teamAffiliations {
      name
    }
  }
}
```


Client

High level

Type safe

???



```
@GraphQLClientApi
interface SuperHeroesApi {
    List<SuperHero> allHeroesIn(String city);
}

class SuperHero {
    private String name;
    private List<String> superPowers;
}

class MyApplication {
    @Inject SuperHeroesApi superHeroesApi;

    // ...
    List<SuperHero> allHeroes =
        superHeroesApi.allHeroesIn("New York");
}
```

```
query {
  allHeroesIn(city: "New York"){
    name
    superPowers
  }
}
```



Other

- @RequestInfo / @RequestContext / ?
- @Pageable
 - offset & limit
- Security
 - Authentication, Authorization
- Other MicroProfile APIs
 - Metrics, Tracing, Fault tolerance, JWT Propagation
- Much more
 - <https://github.com/eclipse/microprofile-graphql/issues>

70

Disclaimer: Work in progress !



THANKS!

Any questions?



You can find me at

www.phillip-kruger.com

 [@phillipkruger](https://twitter.com/phillipkruger)

<https://bit.ly/mp-graphql-presentation-jozijug>
<http://bit.ly/mp-graphql-example>



CREDITS

Special thanks to all the people who made and released these awesome resources for free:

- ◆ Presentation template by [SlidesCarnival](#)
- ◆ Icons by [HandDrawnGoods](#)
- ◆ Code markup with [SlidesCodeHighlighter](#)